Linguaggi e tecnologie per il Web

HTML5

Ing. Massimo Nannini

HTML5: breve storia

- **HTML5** è il successore di HTML 4 e XHTML 1.0, standardizzati dal W3C rispettivamente nel 1999 e 2000
- Nel 2002 il W3C decide che la futura versione 2.0 di XHTML debba rimpiazzare HTML (questo implica la non-retrocompatibilità del futuro linguaggio con HTML 4), ed essere document-oriented e non application-oriented
- Nel 2004 si forma il consorzio WHATWG (Web Hypertext Application Technology Working Group) in opposizione al W3C e alla sua decisione di abbandonare HTML per XHTML
- WHATWG rilascia nel 2008 il primo draft del suo HTML5

HTML5: breve storia (segue)

- Nel frattempo il W3C torna sui suoi passi, e finisce per sposare la visione WHATWG. Il progetto XTML 2.0 viene chiuso, e parte il lavoro per la standardizzazione di HTML5
- Nel 2012 WHATWG rilascia l’”HTML5 living standard”, ovvero una specifica di HTML5 che verrà lasciata evolvere continuamente (senza rilascio di versioni specifiche)
- Nel 2014 il W3C rilascia il suo standard HTML5

HTML5 vs. HTML 4

- Nuove marcature strutturali e «semantiche»
- Altri nuovi tag
- Elementi HTML editabili
- Supporto ai microdati
- Nuovi tag e attributi per le form
- Nuove API

HTML5 vs. HTML4 (segue)

Nuove API create per supportare:

- Multimedialità
- Geolocalizzazione
- Esecuzione asincrona e parallela di script
- Comunicazioni bidirezionali tra web client e web server
- Drag and drop
- Applicazioni web offline
- Grafica
- Cronologia della navigazione

Nuovi Tag HTML5

- <header>
- <footer>
- <section>
- <article>
- <nav>
- <aside>
- <main>
- <mark>
- <time>
- <meter>
- <progress>

Nuovi Tag HTML5

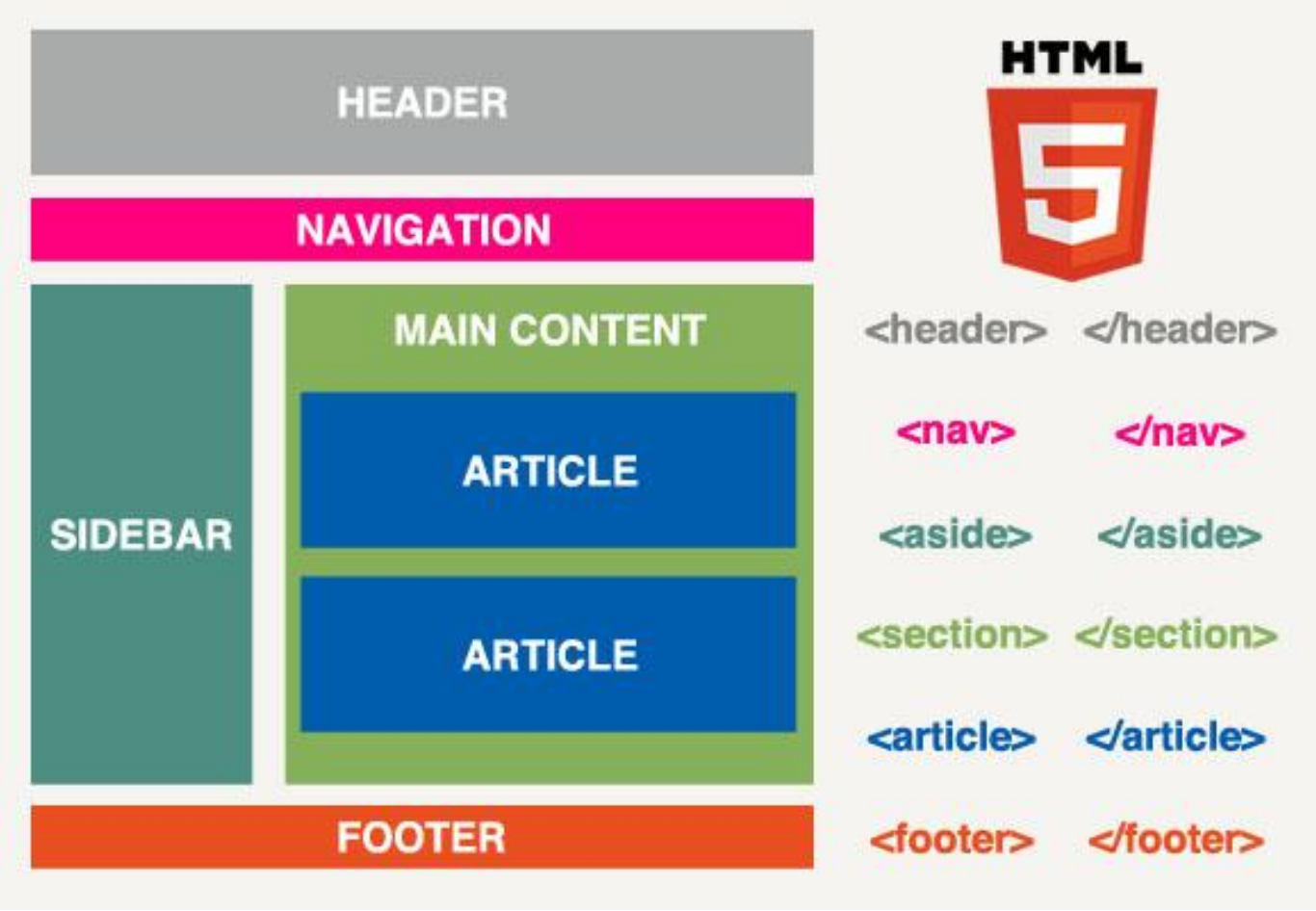
- <figure>
- <figcaption>
- <ruby>
- <wbr>
- <command>
- <menu>
- <details>
- <summary>
- <keygen>
- <output>

Costrutti comuni delle pagine

Prescindendo dal contenuto la maggior parte delle pagine web sono costituite da quattro componenti principali:

- una intestazione per la navigazione
- un articolo nell'area principale dei contenuti
- un riquadro con le informazioni di contorno
- un piè di pagina

Costrutti comuni delle pagine



Costrutti comuni delle pagine

Analizziamo gli elementi semantici:

- header
- nav
- main
- article
- section
- aside
- footer

Creare una intestazione

Se in una sezione di una pagina c'è un gruppo di contenuti introduttivi o di navigazione lo si deve contrassegnare con l'elemento **header**.

In una pagina ci possono essere uno o più elementi **header**, il cui significato dipende dal contesto.

- in cima alla pagina per contenere banner e menù di navigazione
- in una posizione più profonda per esempio come intestazione di un articolo

<header></header>

```
…..
<body>
<header role="banner">
    <nav>
        <ul>
            <li><a href="#gaudi">Barcelona's Architect</a></li>
            <li lang="es"><a href="#sagrada-familia">La Sagrada Família</a></li>
            <li><a href="#park-guell">Park Guell</a></li>
        </ul>
    </nav>
</header>
<body>
…….
```

Esempio 1
Esempio 2

Non è possibile nidificare un elemento *footer* oppure un altro elemento *header* all'interno di un **header**.
Non è possibile nidificare un **header** in un elemento *footer* o *address*.

Contrassegnare la navigazione

Nelle versione precedenti di HTML non esisteva un elemento che rappresentasse esplicitamente una sezione di link di navigazione (menù).
I link di un **nav** possono puntare a:

- elementi interni alla pagina
- ad altre pagine o risorse esterne

NON tutti i gruppi di link presenti in una pagina devono essere contebuti in un **nav**.

<nav></nav>

```
…..
<body>
    <header role="banner">
        <nav role="navigation">
            <ul>
                <li><a href="#gaudi">Barcelona's Architect</a></li>
                <li lang="es"><a href="#sagrada-familia">La Sagrada Família</a></li>
                <li><a href="#park-guell">Park Guell</a></li>
            </ul>
        </nav>
    </header>
</body>
…….
```

Non è possibile nidificare un elemento **nav** all'interno di un elemento *address.*

Esempio 1
Esempio 2

Creare un elemento main

Come abbiamo detto in precedenza le pagine web contengono diverse sezioni: una intestazione, un piè di pagina, in alcuni casi un riquadro con ulteriori informazioni o link ad altri siti.
Tuttavia, solo una parte della pagina rappresenta i suoi contenuti principali. Questi contenuti devono essere racchiusi nell'elemento **main**.
E' uno degli elementi più recenti introdotti da HTML5.

<main></main>

```
…….
<!-- ==== START MAIN ==== -->
<main role="main">
    <article>
        <h1 id="gaudi">Barcelona's Architect</h1>
        <p>Antoni Gaudí's incredible buildings bring millions ...</p>
        ... [rest of main page content] ...
    </article>
</main>
<!-- end main -->
……..
```

Esempio 1

Non è possibile nidificare un elemento **main** all'interno degli elementi *article*, *aside*, *footer*, *header*, *nav*.
Deve essere utilizzato solo una volta in ciascuna pagina.

Creare un elemento article

Basandosi sul suo nome si potrebbe immaginare che **article** si utilizzi per includere contenuti come per esempio un articolo di giornale. In realtà non si limita a questo, in HTML5 il significato di **article** non viene inteso in modo letterale.

article definizione HTML5

L'elemento **article** *rappresenta una composizione completa , o autocontenuta, in un documento, una pagina, un'applicazione o un sito, che, in linea di principio si può distribuire e riutilizzare liberamente. Può trattarsi di un post di un forum, di una voce in un blog, di un articolo di giornale, di un commento inviato da un utente, di un widget o di un qualsiasi oggetto di contenuto indipendente.*

<article></article>

```
……
<main role="main">
        <article>
                <h1 id="gaudi">Barcelona's Architect</h1>
                <p>Antoni Gaudí's incredible buildings bring millions of ...</p>
                <p>Gaudí's non-conformity, already visible in his ....</p>
                <h2 id="sagrada-familia">La Sagrada Família</h2>
                <p><img src="img/towers.jpg" width="75" height="100" alt="Sagrada Família
                Towers" /> The complicatedly named ....</p>
                <h2 id="park-guell">Park Guell</h2>
                <p><img src="img/dragon.jpg" width="100" height="75" alt="Park Guell
                Dragon" /> The Park Guell always reminds me ....</p>
                <p>Perhaps that is for the best, since now we <em>all</em> get to enjoy it.
                The Park Guell ....</p>
        </article>
</main>
…….
```

Esempio 1

Esempio 2

E' possibile nidificare un elemento **article** all'interno di un altro a patto che siano correlati tra loro.
Una pagina può contenere uno o più elementi **article**.

Definire un elemento section

L’elemento **section** rappresenta una sezione generica di un documento o di una applicazione. Una sezione rappresenta un raggruppamento tematico di contenuti, tipicamente con un titolo.
Anche se **section** è definito in parte come sezione "generica" non lo si deve confondere con l’elemento veramente generico *div.*
Dal punto di vista semantico, **section** identifica una sezione separata della pagina, mentre *div* non trasmette nessun significato.

<section></section>

```
<body>
<h1>Graduation Program</h1>
    <section>
        <h2>Ceremony</h2>
            <ol>
                <li>Opening Procession</li>
                <li>Speech by Valedictorian</li>
            </ol>
    </section>
    <section>
        <h2>Graduates (alphabetical)</h2>
            <ol>
                <li>Molly Carpenter</li>
                <li>Anastasia Luccio</li>
            </ol>
    </section>
<body>
```

Esempio 1
Esempio 2

E' possibile nidificare elementi **section** in un *article* per contrassegnare in modo esplicito le sezioni di un capitolo. Se si vuole aggiungere un contenitore per motivi stilistici si deve utilizzare *div*.

Specificare un elemento aside

Quando dobbiamo rappresentare una sezione di contenuti che in qualche modo è correlata con il contenuto che lo circonda, ma che in se è autonoma (concettualmente e non visivamente) si utilizza l'elemento **aside**.

Anche se comunemente si pensa ad **aside** come ad un riquadro laterale, è possibile posizionarlo anche nidificato all'interno del contenuto principale. In questo caso dovrà essere correlato a quel contenuto e non alla pagina in generale.

<aside></aside>

```
main role="main">
        <article>
                <h1 id="gaudi">Barcelona's Architect</h1>
                ......
        </article>
</main>

<!-- ==== START ASIDE ==== -->
<aside role="complementary">
        <h1>Architectural Wonders of Barcelona</h1>
        <p>Barcelona is home to many architectural wonders in addition to Gaudí's work. Include:</p>
        <ul>
                <li lang="es">Arc de Triomf</li>
                <li>The cathedral <span lang="es">(La Seu)</span></li>
                .....
        </ul>
        <p><small>Credit: <a href="http://www.barcelona.de/en/architecture-buildings.html"
        rel="external"><cite>Barcelona.de</cite></a>.</small></p>
</aside>
<!-- end aside -->
```

Esempio 1
Esempio 2
Esempio 3

Non è possibile nidificare elementi **aside** in un elemento *address*.
I contenuti dei riquadri laterali devono essere inseriti dopo il contenuto principale della pagina per motivi di SEO.

Creare un elemento footer

Normalmente si pensa al piè di pagina, ma **footer** oltre ad essere assolutamente adatto a questo scopo più essere utilizzato anche altrove, come già *header*.
L'elemento **footer** rappresenta il piè di pagina del più vicino elemento *article*, *aside*, *blockquote*, *boby*, *details*, *fields-set*, *figure*, *nav*, *section* o *td* nel quale è nidificato.
E' il piede dell'intera pagina solo quando il suo elemento più vicino è *body.*

<footer></footer>

```
<body>
        <header role="banner">
                <nav role="navigation">
                .....
                </nav>
        </header>

        <main role="main">
                <article>
                ....
                </article>
        </main>

        <aside>
                ....
        </aside>

        <!-- ==== START PAGE FOOTER ==== -->
        <footer role="contentinfo">
                <p><small>&copy; Copyright All About Gaudí</small></p>
        </footer>
        <!-- end page footer -->
</body>
```

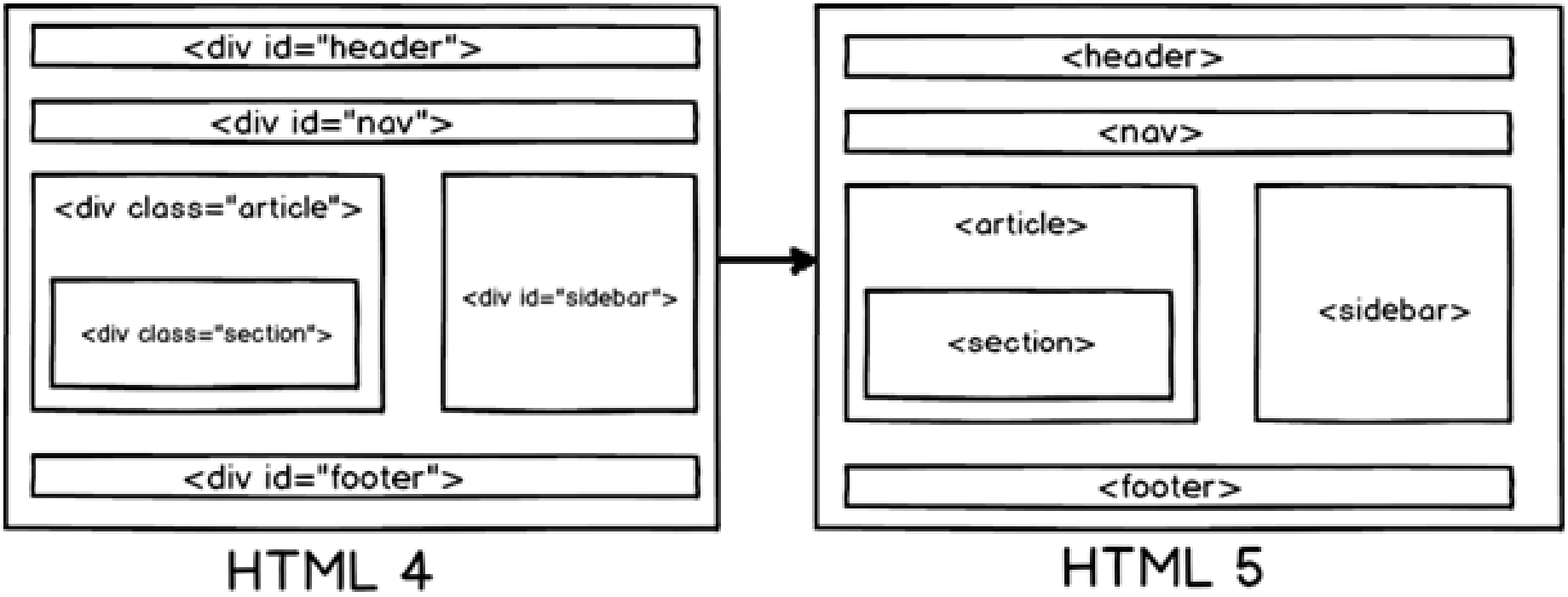
Esempio 1
Esempio 2
Esempio 3

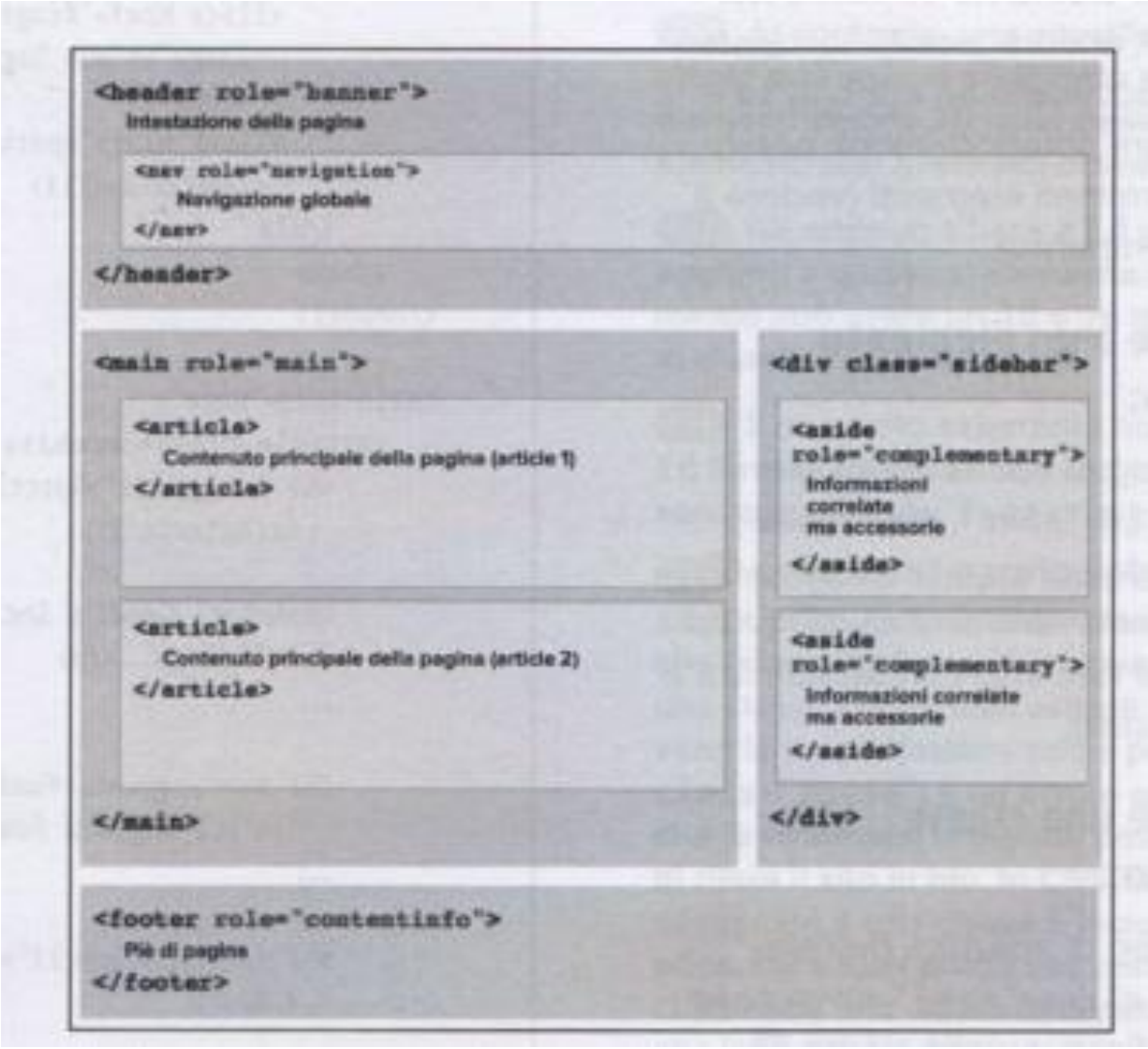
Non è possibile nidificare elementi **footer** in un elemento *address*.
Non è possibile nidificare elementi **footer** in altri elementi **footer**.
Non è possibile nidificare elementi *header* in un **footer**.
Non è obbligatorio che il footer sia

Il contenitore generico div

A volte è necessario racchiudere un segmento di contenuto per applicarvi alcune formattazioni con i CSS3 o un effetto con JavaScript.
L'elemento **div** esisteva anche prima di HTML5 ed era la scelta obbligata per racchiudere contenuti quali intestazioni, piè di pagina, contenuto principale, ecc.., **non ha però un significato semantico**.

HTML4 vs. HTML5





[Esempio 1]
[Esempio 2]
[Esempio 3]

L’elemento spam

L’elemento **spam** come *div* è del tutto privo di **significato semantico**.
La differenza è che **span** è appropriato soltanto intorno ad una parola o a una frase, mentre *div* è per blocchi di contenuto.

<spam></spam>

```
<!DOCTYPE html><html lang="en">
<head>
<meta charset="UTF-8" />
        <title>Creating Spans</title>
</head>
<body>
        <h1 lang="es">La Casa Milà</h1>
        <p>Gaudí's work was essentially useful. <span lang="es">La Casa Milà</span> is an apartment
        building and <em>real people</em> live there.</p>
</body>
</html>
```

Gli elementi figure e figcaption

L'elemento **figure** rappresenta un blocco distinto dal testo principale, pensato per contenere immagini, diagrammi, grafici, tabelle, esempi di codice, ecc. A **figure** possiamo opzionalmente associare una didascalia o una descrizione tramite **figcaption**.
Prima di HTML5 non esisteva questo elemento dedicato per cui si era costretti ad utilizzare elementi *div* non semantici.

<figure></figure> <figcaption></figcaption>

```
<body>
      <article>
      <h1>2013 Revenue by Industry</h1>
      <p>. . . [report content] . . .</p>
      <figure>
            <figcaption><b>Figure 3:</b> Breakdown of Revenue by Industry</figcaption>
            <img src="img/chart-revenue.png" width="180" height="143" alt="Revenue chart: Clothing 42%,
            Toys 36%, Food 22%" />
      </figure>
      <p>As Figure 3 illustrates, . . . </p>
      <p>. . . [more report content] . . .</p>
      </article>
</body>
```

Per impostazione predefinita i browser applicano agli elementi **figure** un margine di 40px a sinistra e a destra. Questi possono essere modificati attraverso le proprietà CSS3 *margin-left:xx* e *margin-right:xx*.
Inoltre per fare scorrere il testo intorno al bordo destro o sinistro si usa *figure* {float:left;} oppure *figure* {float:right;}

Esempio 1
Esempio 2

Altri elementi HTML5

Analizziamo ora altri elementi introdotti da HTML5 per i testi:

- mark
- time
- meter
- progress
- ruby
- wbr

L’elemento mark

E’ l’elemento evidenziatore di HTML5, infatti può essere considerato la versione semantica del pennarello evidenziatore.
Lo scopo è quello di mettere in evidenza determinate parole per attrarre l’attenzione del lettore su quel determinato segmento di testo.

<mark></mark>

```
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="UTF-8" />
	<title>Highlighting Text</title>
</head>
<body>
	<p>So, I went back and read the instructions myself to see what I'd done wrong. They said:</p>
	<blockquote>
		<p>Remove the tray from the box. Pierce the overwrap several times with a fork and cook on
		High for <mark>15 minutes</mark>, rotating it half way through.</p>
	</blockquote>
	<p>I thought he'd told me <em>fifty</em>. No wonder it exploded in my microwave.</p>
</body>
</html>
```

L'elemento **mark** è diverso da *em* (enfasi) e da *strong* (importanza)
Per i browser più datati oppure per cambiare colore è necessario applicare uno stile *mark* {backgroud-color:yellow;}

Esempio 1
Esempio 2

L’elemento time

L’elemento **time** rappresenta una data e/o un orario nel calendario Gregoriano.
Il testo contenuto all’interno di **time** è quello che appare sullo schermo, mentre il valore dell’attributo **datetime** serve per fornire informazione al computer sulla data/ora indicata.
Il formato da utilizzare è: YYYY-MM-DDThh:mm:ss quando no è specificata la proprietà **datetime**.

<time *datetime=" "*></time>

```
<body>
    <p>The train arrives at <time>08:45</time> and <time>16:20</time> on <time>2017-03-19</time>.</p>
    <p>They made their dinner reservation for <time datetime="2013-11-20T18:30:00">tonight at
    6:30</time>.</p>
    <p>We began our descent from the peak of Everest on <time datetime="1952-06-12T11:05:00">June 12,
    1952 at 11:05 a.m.</time></p>
    <p>The film festival is <time datetime="2014-07-13">July 13</time>-<time datetime="2014-07-
    16">16</time>.</p>
    <!-- Example with no year -->
    <p>Her birthday is <time datetime="03-29">March 29th</time>.</p>
    <!-- Example of durations -->
    <p>The meeting lasted <time>2h 41m 3s</time> instead of the scheduled <time datetime="2h
    30m">two hours and thirty minutes</time>.</p>
</body>
```

Esempio 1
Esempio 2

L'attributo *datetime* di per sé non fa nulla ma può essere utilizzato per sincronizzare date e orari tra appliazioni (es. app calendario)

L’elemento meter

L’elemento **meter** si utilizza per indicare un valore frazionale oppure una misura in un intervallo noto sotto forma di barra colorata.
L’elemento supporta alcuni parametri:

- *high*, *low* per suddividere l’intervallo in tre zone basso color=green, medio color=yellow, alto color=red
- *min*, *max* per specificare i valori massimo e minimo consentiti (se omessi hanno valore 0 e 1.0)
- *optimun* per indicare il valore ottimale
- *value* per indicare il valore attuale della misura (obbligatorio)

<meter></meter>

```
<body>
	<p>Project completion status: <meter value="0.80">80% completed</meter></p>
	<p>Car brake pad wear: <meter low="0.25" high="0.75" optimum="0" value="0.80">21% worn</meter></p>
	<p>Miles walked during half-marathon: <meter min="0" max="13.1" value="5.5"
	title="Miles">4.5</meter></p>
</body>
```

Esempio 1

Il testo compreso tra <meter>*text*</meter> non viene visualizzato salvo nel caso in cui il browser non supporti l'elemento.

L’elemento progress

L’elemento **progress** si utilizza per indicare lo stato di progressione di una operazione sotto forma di barra colorata (es. barra di caricamento).
L’elemento supporta alcuni parametri:

- *max* specifica il valore massimo raggiungibile (es. 100)
- *form* corrisponde all’id di un elemento form presente nella pagina in cui progress non è nidificato
- *Value* specifica il valore corrente

<progress></progress>

```
<body>
    <p>Please wait while we save your data.</p>
    <p>Current progress: <progress max="100" value="0">0% saved</progress></p>
</body>
```

Esempio 1
Esempio 2

Il testo compreso tra <progress>*text*</progress> non viene visualizzato salvo nel caso in cui il browser non supporti l'elemento.

Aggiornamento di *value* in meter e progress

```
<!DOCTYPE html>
<html lang="en">
  <head>
        <script type="text/javascript">
                var bar = document.getElementById('progressBar');
                bar.value = 50;
        </script>
  </head>
  <body>
        <progress id="progressBar" value="0" max="100">0% saved</progress>
  </body>
</html>
```

L’elemento ruby, rp e rt

Le *annotazioni* ***ruby*** sono una convenzione nelle lingue dell’Asia orientale, come il cinese e il giapponese e vengono utilizzate per mostrare la pronuncia degli ideogrammi.
All’interno dell’elemento **ruby**, tramite il tag figlio **rt** (*ruby text*), possiamo associare a ciascun ideogramma un testo esplicativo che può essere separato dal resto tramite parentesi incluse nei tag rp (*ruby parenthesis*).

<ruby></ruby>

```
<body>
<ruby>
      北 <rp>(</rp><rt>ㄅㄟˇ</rt><rp>)</rp>
      京 <rp>(</rp><rt>ㄐㄧㄥ</rt><rp>)</rp>
</ruby>
</body>
```

Esempio 1
Esempio 2

L’elemento wbr

L’elemento **wbr** è un "cugino" di *br* (*interruzione di riga*) e rappresenta una interruzione di riga opzionale.
La riga di testo viene spezzata su due righe in base al layout della pagina e alla larghezza del tag contenitore.

```
<p>
    They liked to say, "FriendlyFleasandFireFlies<wbr />
    FriendlyFleasandFireFlies<wbr />
    FriendlyFleasandFireFlies<wbr />
    " as fast as they could over and over.
</p>
```

Esempio 1
Esempio 2

Nuovi elementi interattivi

Analizziamo ora altri elementi introdotti da HTML5 di tipo interattivo:

- details
- summary
- menu
- command (non ancora implementato!!)

Nuovi elementi interattivi

Questi elementi consentono di raggruppare controlli che permettono di eseguire azioni sulla pagina senza la necessità di eseguire un round-trip verso il server, come avviene nel caso delle form.
Nella lista precedente è presente l'elemento *menu* che in HTML4 era stato deprecato, ma in HTML5 assume una nuova connotazione e permette di definire una lista o un menù di comandi.

Gli elementi details e summary

Rappresenta un insieme di informazioni o controlli aggiuntivi che possono essere mostrati a richiesta nella pagina.
Può includere opzionalmente come primo elmento figlio un tag **summary**, il cui scopo è quello di riportare una informazione riepilogativa che descriva il contenuto di dettaglio.

<details></details> <summary><summary>

```
<section>
        <hgroup>
                <h1>HTML5</h1>
                <h2>Capitolo 2 - Nuovi elementi del markup</h2>
        </hgroup>
        <details open>
                <summary>Download del file in corso...</summary>
                <ul>
                        <li>Nome file: capitolo2.pdf</li>
                        <li>Tipologia: PDF</li>
                        <li>Dimensione: 200 KB</li>
                </ul>
        </details>
</section>
```

Esempio 1

Impiego dell'elemento details

L'elemento **details** prevede un attributo booleano *open*, che se presente rende il contenuto del tag visibile.

Uno degli utilizzi più interessanti consiste nell'utilizzarlo come contenitore di controlli che debbano essere resi visibili sono in alcune situazioni o dipendentemente da input dell'utente utilizzando appunto l'attributo *open.*

```
<script type="text/javascript">
        document.getElementById("detId").setAttribute("open", "open");
</script>
```

L’elemento menu

L’elemento **menu** consente di realizzare toolbar e menù contestuali all’interno delle pagine e deve essere utilizzato come contenitore per raggruppare in modo strutturato diversi comandi.

Il tag presenta due attributi:

- *label:* che consente di indicare una etichetta testuale
- *type*: che permette di definire la tipologia del menù
  - context: l’elemento menù rappresenta un menù contestuale
  - toolbar: l’elemento menù rappresenta una toolbar
  - list: l’elemento menù rappresenta una semplice lista non ordinata di elmenti

ATTENZIONE: attualmente supportato solo da FireFox

<menu></menu>

```
<menu type="context" id="mymenu">
  <menuitem label="Refresh" onclick="window.location.reload();" icon="ico_reload.png"></menuitem>
  <menu label="Share on...">
    <menuitem label="Twitter" icon="ico_twitter.png"
onclick="window.open('//twitter.com/intent/tweet?text=' + window.location.href);"></menuitem>
    <menuitem label="Facebook" icon="ico_facebook.png"
onclick="window.open('//facebook.com/sharer/sharer.php?u=' + window.location.href);"></menuitem>
  </menu>
  <menuitem label="Email This Page"
onclick="window.location='mailto:?body='+window.location.href;"></menuitem>
</menu>
```

I nuovi attributi

Oltre ai nuovi tag introdotti, HTML5 introduce una serie di attributi aggiuntivi per gli elementi già esistenti in HTML4 e/o nuovi HTML5.
La maggior parte delle novità riguardano gli elementi che sono utilizzati all'interno delle form come *input*, *textarea* e *button*.

Attributo	Elementoli
async	<script>
autocomplete	<input>
autofocus	<input>, <select>, <textarea>, <button>
charset	<meta>
disabled	<fieldset>
placeholder	<input>, <textarea>
dirname	<input>, <textarea>
form	<input>, <select>, <textarea>, <button>, <fieldset>
formaction	<input>, <button>
formenctype	<input>, <button>
formmethod	<input>, <button>
formnovalidate	<input>, <button>
hreflang	<area>
label	<menu>
list	<input>
manifest	<html>
max	<input>
media	<a>, <area>
min	<input>
multiple	<input>
novalidate	<form>
pattern	<input>
rel	<area>
required	<input>, <textarea>
reserved	
sandbox	<iframe>
scoped	<style>
seamless	<iframe>
sizes	<link>
srcdoc	<iframe>
start	
step	<input>
target	<a>, <area>, <base>
type	<menu>
value	
start	

Supporto ai microdati

- i microdati servono a creare un tagging «semantico» di porzioni del documento
- sono basati vocabolari che definiscono identificatori di proprietà relative ad un (micro-)dominio di interesse
- si utilizzano gli attributi HTML itemscope, itemtype e itemprop

Esempio: il vocabolario Person

Property	Description
name (fn)	Name.
nickname	Nickname.
photo	An image link.
title	The person's title (for example, Financial Manager).
role	The person's role (for example, Accountant).
url	Link to a web page, such as the person's home page.
affiliation (org)	The name of an organization with which the person is associated (for example, an employer). If fn and org have the exact same value, Google will interpret the information as referring to a business or organization, not a person.
friend	Identifies a social relationship between the person described and another person.
contact	Identifies a social relationship between the person described and another person.
acquaintance	Identifies a social relationship between the person described and another person.
address (adr)	The location of the person. Can have the subproperties street-address, locality, region, postal-code, and country-name.

Microdati in HTML5: esempio

```
…
<div itemscope itemtype="http://datavocabulary.org/Person">
Benvenuti nella home page di
<span itemprop="name">Riccardo Rosati</span>,
<span itemprop="title">professore associato</span> alla
<span itemprop="affiliation">Sapienza Università di
Roma</span>.
</div>
…
```

Modificare il contenuto di una pagina

I seguenti nuovi attributi globali permettono di dichiarare elementi del documento HTML modificabili da utente:

- contenteditable
- contextmenu
- data-* (attributi definibili da utente)
- draggable
- hidden
- spellcheck

Nuove API

- HTML5 aumenta significativamente le API messe a disposizione degli script
- Gli obiettivi sono:
  - In generale, accrescere le possibilità offerte alla programmazione lato client
  - Standardizzare alcuni tipi più comuni di operazioni effettuate lato client
  - Aggiornare le API alle necessità dei media agent più recenti (smartphone, tablet)

Nuove API

- Gestione di risorse e flussi audio e video
- Accesso off-line alle applicazioni
- Estensione delle capacità di comunicazione, sia verso il web server che verso altre applicazioni
- Esecuzione di azioni in background
- Estensione del concetto di cookie (salvataggio di informazioni sul dispositivo dell'utente)
- Gestione della cronologia della navigazione

Nuove API (segue)

- Text editing
- Gestione del «drag and drop»
- Generazione di oggetti grafici 2D/3D
- Gestione di informazioni multimediali generate dall'utente (ad esempio mediante webcam e microfono)

Offline API

- permettono di salvare copie locali di un insieme di risorse, allo scopo di permettere al browser di eseguire applicazioni anche in modalità offline
- l'elenco delle risorse da salvare è contenuto in un file chiamato **manifest** (MIME type 'text/cache-manifest')
- L'attributo manifest del tag html permette di dichiarare il file manifest associato al documento HTML
- La cache costituita dalle risorse elencate nel file manifest è gestita da apposite API (associate all'oggetto corrispondente alla proprietà applicationCache dell'oggetto window)

WebStorage API

- Estendono le capacità di memorizzazione dei cookies
- Oggetti **localStorage** e **sessionStorage** (accessibili come array associativi)
- Possono memorizzare solo stringhe (testo): per memorizzare oggetti arbitrari occorre serializzarli (ad esempio tramite JSON)

WebSocket API

- Permettono di instaurare una connessione dati bidirezionale tra web client e web server
- La creazione di un nuovo oggetto WebSocket crea una connessione con un server (la cui url va specificata nel costruttore)
- La funzione associata all'event handler onmessage viene eseguita quando dal server arriva un messaggio
- Il metodo send(x) invia il testo x al server

Canvas

- elemento <canvas> per dichiarare una zona della pagina su cui è possibile disegnare, tramite nuove API
- si può disegnare una canvas in un contesto 2D o in un contesto 3D
- le API per disegnare (in contesto 2D) si dividono in
  - metodi path (linee, archi,…)
  - metodi modificatori (rotazioni, traslazioni,…)
  - metodo drawImage (disegna un'immagine)
  - metodi per scrivere testo
  - metodi per scrivere singoli pixel

Geolocation API

Servono a gestire dati geospaziali, tipicamente la posizione (anche se questa è effettivamente disponibile solo su alcuni user agent)

La proprietà geolocation dell'oggetto navigator ha due metodi per ottenere la posizione corrente:

- **getCurrentPosition**
- **watchPosition**

(il secondo metodo differisce dal primo perché restituisce una nuova posizione ogni volta che questa cambia)

Audio/Video e nuove API

- HTML5 permette la gestione nativa (ovvero senza ricorrere a plug-in del browser) di contenuti multimediali
- tag **<video>**: permette di inserire un contenuto video
- tag **<audio>**: permette di inserire un contenuto audio
- il formato dei video e degli audio supportati dipende dai browser, tuttavia esistono dei formati di riferimento (mp4, webm, ogg per i video)
- esistono delle nuove API per video e audio che permettono la gestione di questo tipo di risorse da script

Form

- Nuovi attributi ed input types per le form sono stati introdotti in HTML5
- L'obiettivo principale è quello di permettere di definire le più comuni forme di validazione di form lato client direttamente nel documento HTML (cioè senza bisogno di aggiungere codice JavaScript)
- Sono stati inoltre aggiunti tipi di elementi utili soprattutto nei nuovi media agent (smartphone e tablet)

Autofocus, placeholder, form

Nuovi attributi:

- **autofocus** (booleano): all'apertura della form, il focus va sull'elemento che ha dichiarato autofocus
- **placeholder** (per elementi input o textarea): valore che all'apertura della form compare sul campo editabile (N.B.: non corrisponde ad un valore (value) iniziale del campo, viene solo visualizzato all'inizio)
- **form**: attributo che permette di associare un elemento ad una form. E' così possibile, ad esempio, dichiarare un campo che appartiene a più form, o dichiarare un campo all'esterno di un elemento form

Required, autocomplete

- **required** (booleano): rende obbligatoria la compilazione dell'elemento al momento del submit della form a cui l'elemento appartiene
- **autocomplete**: attributo che può assumere due valori:
  - on = il browser può effettuare l'**autocompletion** di questo campo, cioè completare il campo in maniera automatica usando i valori precedentemente inseriti in questo campo
  - off = il browser non può fare l'autocompletion
  - se autocomplete non viene assegnato, viene usato il default del browser (di solito è on)

Multiple, pattern

- **multiple** (booleano): permette al campo di accettare valori multipli
- esempio:

```
<form action="demo_form.asp">
  Select images: <input type="file" name="img" multiple>
  <input type="submit">
</form>
```

- **pattern**: viene assegnato ad una espressione regolare; i valori del campo devono appartenere al linguaggio denotato dall'espressione regolare

Min, max, step

- **min** = valore minimo ammesso
- **max** = valore massimo ammesso
- **step** = intervallo tra un valore ammesso e il successivo
- **novalidate** (booleano): se dichiarato sull'elemento form, indica che **non** verrà effettuata la validazione degli elementi di quella form

Nuovi input types

I seguenti input types permettono di gestire informazioni testuali di tipo specifico:

- tel
- search
- url
- email

Nuovi input types (segue)

- color: permette la selezione di un colore
- number: permette l'inserimento di un numero
- range: permette l'inserimento di un numero attraverso uno slider (cursore orizzontale)

Nuovi input types (segue)

Per gestire le date sono stati introdotti i seguenti input types:

- datetime
- datetime-local
- date
- month
- week
- time

Il tag <datalist>

- Permette di definire un campo testuale sul quale il browser può effettuare autocompletion usando un insieme predefinito di valori (l'utente però è libero di scrivere un valore al di fuori dell'elenco predefinito)
- esempio:

```
<input type="text" name="giudiz"list="listagiudizi">
    <datalist id="listagiudizi">
      <option value="insufficiente">
      <option value="sufficiente">
      <option value="discreto">
      <option value="buono">
      <option value="ottimo">
    </datalist>
```